EXPRESS MAIL NO. EV020611911US

(19) Europäisches Patentamt
European Patent Office
Office européen d s brevets

(11) Publication number : **0 569 312 A2**

(12) EUROPEAN PATENT APPLICATION

(21) Application number : **93480035.0**

(22) Date of filing : **06.04.93**

(51) Int. Cl.[5] : **G06F 9/28**, G06F 9/38

(30) Priority : **06.05.92 US 879651**

(43) Date of publication of application :
10.11.93 Bulletin 93/45

(84) Designated Contracting States :
DE FR GB

(71) Applicant : **INTERNATIONAL BUSINESS MACHINES CORPORATION**
Old Orchard Road
Armonk, N.Y. 10504 (US)

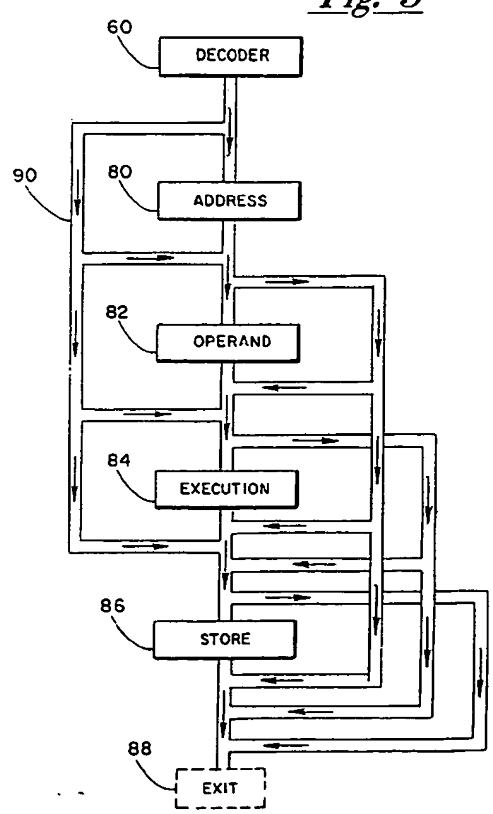
(72) Inventor : **Carnevale, Michael Joseph**
2103 18 1/2 Avenue N.W., No. 4
Rochester, Minnesota 55901 (US)
Inventor : **Kalla, Ronald Nick**
RR No. 1, 77A East Ryans Bay Road
Zumbro Falls, Minnesota 55991 (US)
Inventor : **McClannahan,Paul Gary**
IBM Corp. Dept 49J 3605 Highway 52 North
Rochester,Minnesota 55901 (US)
Inventor : **Trombley, Michael Raymond**
109 Meadow Run Drive S.W.
Rochester, Minnesota 55902 (US)

(74) Representative : **Vekemans, André**
Compagnie IBM France Département de Propriété Intellectuelle
F-06610 La Gaude (FR)

(54) Variable stage entry/exit instruction pipeline.

(57) An instruction pipeline includes a sequence of interconnected pipeline stages (80, 82, 84, 86), each stage dedicated to one of several operations executed on data in a digital processing device. Control words govern execution of the operations as they progress through the pipeline. The pipeline stages, as well as the pipeline entry and exit (88), are interconnected in a manner that permits each control word to enter and exit the pipeline at any one of the stages, and to skip any stages in which the control word will not govern any operations on data. On occasion, this permits a control word to bypass another control word which originally preceded it in the pipeline, thus to reverse the order of the two control words. A mapping field in each control word predetermines its route through the instruction pipeline, one bit of the map field corresponding to each pipeline stage. The route of each control word further is controlled by arbitration logic to resolve contentions among control words for particular pipeline stages, and to insure against a reversal in control word order when such reversal might produce an error. Each pipeline stage is a data register configured to accommodate one control word at a time. Succeeding stage registers are selectively reduced in length to eliminate selected segments and bit positions, thereby deleting from each control word certain bits and multiple bit fields executed in preceding stages.

Fig.-3



EP 0 569 312 A2

Jouve, 18, rue Saint-Denis, 75001 PARIS

The present invention relates to pipelined hardware for sequentially handling instructions that govern data operations in digital processors, and more particularly to a means for selectiv ly and individually routing instructions through the pipelined hardware.

One known manner of improving data processing speeds in devices for processing digital data, is to configure each computer program instruction as a series of microinstructions presented to a pipeline, and then to perform operations on the microinstructions (or on other data according to the microinstructions) as they progress sequentially through a sequence of pipeline stages in the pipeline. This facilitates concurrent execution of different instructions, one instruction residing in each pipeline stage. An example of a pipelined processor is disclosed in U.S. Patent No. 4,450,525 (Demuth et al).

While the configuration of microinstructions can vary from one system to another, each microinstruction typically includes a set of multiple bit fields containing control information used by system hardware, including pipelined hardware. Each of the control fields is specifically related to one of the pipeline stages, in that the control field is valid only when the microinstruction resides in the related stage. Particular microinstructions can include one or more control fields relating to one, or all, of the pipeline stages.

The pipeline stages are connected in sequence between a pipeline entry and a pipeline exit. Accordingly, each of the microinstructions progresses sequentially through every pipeline stage, regardless of the number of pipeline stages to which it relates. Consequently, a substantial number of machine cycles are consumed in which microinstructions reside in unrelated pipeline stages, with no operations performed, either on the microinstructions themselves or other data within the processing device.

Therefore, it is an object of the present invention to provide a multiple stage instruction pipeline configured for more effective use of available machine cycles, reducing the number of cycles in which microinstructions reside in unrelated pipeline stages.

Another object is to provide a process for selectively routing microinstructions or control words through a multiple stage instruction pipeline, whereby each microinstruction or control word temporarily resides only in selected stages of the pipeline.

A further object is to provide each of a plurality of microinstructions with a mapping field that identifies the data operations, and therefore the stages, to which the microinstruction relates.

Yet another object of the invention is to provide a means for arbitration among control words or microinstructions competing for access to a particular pipeline stage as they progress through an instruction pipeline.

To achieve these and other objects, there is provided an apparatus for governing the execution of operations on data in a digital processing device. The apparatus includ s a pipeline having a plurality of pipeline stages. Each pipeline stage is dedicat d to one of a plurality of data operations executed on data in the processing device. A pipeline input receives digital control words into the pipeline. The digital control words govern the execution of the data operations as they progress through the pipeline. Each of the control words includes control information corresponding to at least a selected one of the data operations, and mapping information that identifies each of the selected data operations. Each control word further governs execution of each selected data operation when residing in the pipeline stage dedicated to that data operation. The apparatus further includes a pipeline exit, and means connecting the pipeline input and exit to each of the pipeline stages, whereby control words can enter and leave the pipeline at any one of the pipeline stages. A control word routing means selectively and individually routes each of the control words through the pipeline depending upon the mapping information of the control word.

Preferably the pipeline stages are connected sequentially, with each control word bypassing stages other than the selected pipeline stages. Stages are selected based upon the mapping information, in the sense that the control word is routed through a pipeline stage only if the word includes control information corresponding to the data operation of that particular stage. Each control word can have a mapping field with a plurality of bit positions, one bit position corresponding to each pipeline stage. A binary "1" in a given bit position indicates that the control word includes control information relating to that pipeline stage. Accordingly, the mapping field identifies all pipeline stages relating to the control word, and provides a road map for routing the control word only through the related stages.

The result is more effective use of available machine cycles in the pipeline. Instructions can enter the pipeline at the first stage in which the instruction is to execute a data operation, rather than at the initial pipeline stage. Similarly, an instruction can exit the pipeline at the final pipeline stage of its execution, rather than at the final stage of the pipeline. An instruction (i.e. a control word) can bypass pipeline stages during its execution. This allows a particular instruction to "leap frog" another instruction initially ahead of it in the pipeline, so long as the intermediate pipeline stages are not used by the particular instruction. A substantial improvement in performance is realized, because more instructions can be executed during the same number of availabl machin cycles.

Th ability of control words to bypass pip line stages gives rise to potential conflicts among two or more control words attempting to ent r one of the pipeline stages. Accordingly, arbitration logic is provided to resolve such contentions. A pip line configured

in accordance with th present invention thus calls for logic circuitry not needed in conventional, sequentially coupled pipeline stages. However, the additional logic can be configured on the processor chip, and is more than justified by the performance gain.

A further aspect of the present invention is a process for routing control words or microinstructions through a multiple stage instruction pipeline. The process includes the following steps:

a. identifying, in connection with each control word, the pipeline stages related to the control word in the sense that the control word is to be executed from related stages; and

b. responsive to the identification of the related pipeline stages, selectively routing the control word through the pipeline along a path that includes only the related pipeline stages.

The process can include the further step of modifying each control word as it progresses through the pipeline, by deleting control information relating to each pipeline stage before the control word enters the next subsequent pipeline stage. In the preferred implementation of this process, each of the pipeline stages is a register having latch positions corresponding to a plurality of multiple bit control fields, each control field corresponding to one of the pipeline stages. In this context, the subsequent stage register is configured without the latch positions corresponding to the control fields of previous stages. As a result, information no longer useful is effectively deleted as the control word progresses through the pipeline. Thus, logic (hardware) and physical space requirements are reduced as compared to conventional pipeline stage arrangements, as succeeding registers require fewer latch positions.

Thus in accordance with the present invention, the efficiency of a multiple stage instruction pipeline is substantially enhanced, with microinstructions being selectively routed only to pipeline stages in which they govern operations performed on processor data.

For a further appreciation of the above and other features and advantages, reference is made to the following detailed description and to the drawings, in which:

Figure 1 is a schematic view of an information processing system including a digital processing device, a main storage memory and an interface between the device and memory;

Figure 2 is a block diagram of certain components of the processing device including an instruction pipeline;

Figure 3 is a schematic view illustrating flow of data in the instruction pipeline;

Figure 4 is a schematic view of the instruction pipeline including arbitration means;

Figure 5 is a flow chart illustrating operation of the arbitration logic;

Figur 6 illustrates the format of a digital microinstruction used in the pipeline;

Figur s 7-10 illustrate s gments of the microinstruction in great r detail;

Figure 11 is a schematic view of pipeline stage registers; and

Figure 12 is a schematic view of an alternative instruction pipeline, with an array in lieu of a decoder at the pipeline input.

Turning now to the drawings, there is shown in Figure 1 an information processing network 16 for storing and performing various operations upon bit-encoded data. The network includes a processing device 18, a main storage memory 20 and an interface 22 coupling the processing device and main memory. A data line 24 connects processing device 18 to the interface, while data line 26 connects the interface to main storage memory 20.

Processing device 18 executes assembly level computer program instructions sequentially in performing logical operations on data, for example store, add, and move-character functions. To this end, the processing device employs a horizontal microcode (HMC) for interpreting the assembly level instructions, i.e. decomposing the assembly level instructions into simpler HMC microinstructions or control words that govern the state of the hardware.

Processing device 18 includes a control store 28 containing the horizontal microcode (HMC) and a control store output register 30. The processing device further includes arithmetic execution logic 32 for performing operations on data, and an I/O memory interface 34 for transmitting data to and from main storage memory 20.

Figure 2 illustrates hardware components of the arithmetic execution logic, and paths for flow of data among the components. Data from the main memory can be provided to a variety of destinations via interface 34 as indicated at 36, including first and second multiplexers 38 and 40 associated with an arithmetic execution unit 42 through respective registers 44 and 46; a program counter 48; an instruction assembly buffer 50; and a general register multiplexer 52 providing its output to a set of general purpose registers 54. The interface 34 also can provide data to a status register 56 and a condition code register 58.

A decoder 60 receives the output of assembly buffer 50, decomposing the output into control words or microinstructions in horizontal microcode. The control words are provided to an instruction pipeline 62, where they progress sequentially through one or more pipeline stages of the pipeline and govern operations executed upon data in processing device 18.

The output of instruction pipeline 62 can be provided to multiplexer 38, multiplexer 40, a first address multipl xer 64 and an effectiv address calculating unit 66. Address unit 66 can provide its output t a multipl xer 67 and then to a register 68 that provides

its output to a memory data multiplexer 70, to I/O interface 34, to program counter 48, and to multiplexer 52.

The output of execution unit 42 can be provided: to memory data multiplexer 70 from a register 72; to general register multiplexer 52; and to status register 56 and condition code register 58. The output of the general purpose register set 54 is available to multiplexers 38 and 40, address multiplexer 64 and a further address multiplexer 74, and a multiplexer 76 that provides its output to a decrementing/incrementing unit 78. Unit 78 can provide its output to address multiplexer 74 and general register multiplexer 52. Thus, multiple paths are available for transmission of data among the components in Figure 2 . Actual data transmission does not occur indiscriminately along all available paths, but rather is controlled by the microinstructions as is later explained.

In Figure 3, instruction pipeline 62 is illustrated in greater detail to show the flow of control words through the pipeline. Pipeline 62 includes four stages including an address generation stage 80, an operand loading stage 82, an execution stage 84 and a result storing stage 86. Decoder 60 provides the entry for all control words into the pipeline. All data leaving the pipeline proceeds to one or more of the above-mentioned possible destinations for the instruction pipeline output. A pipeline exit is shown in broken lines at 88. This does not represent an actual hardware component. Rather, exit 88 signifies the point at which the instructions or control words, having served their purpose in performing operations along one or more of the pipeline stages, are no longer needed and leave the pipeline. All control words proceed through pipeline 62 in the same direction (downward as viewed in the figure), but not sequentially in the sense of all microinstructions occupying each stage. Rather, the data pathways interconnecting the stages, indicated generally at 90, provide fifteen distinct routes through pipeline 62, based on the available combinations of one, two, three or all stages.

One feature of the present invention is the interconnection of the pipeline stages, pipeline entry and pipeline exit to provide a variety of routes through the instruction pipeline. In general, for a pipeline composed of n separate stages, the interconnection affords a potential number of routes through the instruction pipeline of 2^n-1. Control words can enter instruction pipeline 62 at any one of the stages, can exit the pipeline at any stage, and can bypass one or more of the stages as they progress through the instruction pipeline. Information in each control word predetermines the route traversed by the control word through the instruction pipeline, as explain d in connection with Figure 6.

While pathway interconnection 90 affords multiple alternative control word routes, it also cr ates the need for arbitration among control words contending for access to particular stages. The need is most apparent in connection with result storing stage 86, configured to receiv control words from decoder 60 and from any one of preceding pipeline stages 80, 82 and 84. Accordingly, an arbitration scheme is provided as shown in Figure 4, which also illustrates the pipeline stages and interconnection 90 in greater detail.

Each of the pipeline stages includes a pipeline register, in particular an address generation register 92, an operand loading register 94, an execution register 96 and a result storing register 98. Each register has individual bit positions and multiple bit segments corresponding to individual binary bits or fields of the control words, whereby each control word temporarily and exclusively occupies one or more of the stage registers when progressing through the instruction pipeline. Address generation register 92 includes a map segment 100, a condition-codes affected bit 102, a valid indication bit position 104, an instruction dependent field segment 106, and four control segments 108, 110, 112 and 114. Each of the control segments, like each of the pipeline stages, is dedicated to a specific type of operation performed on data as the control words progress through the instruction pipeline. For example, control segment 108 stores only the field of a control word that specifically relates to address generation. More than one such field may reside in control segment 108 if the particular control word contains more than one address generation field. The remaining control segments of register 92 are similarly limited to one type of field while adapted to accommodate more than one field of that type. Remaining pipeline stage registers 94, 96 and 98 have similar segments and bit positions, but vary as will be explained in connection with Figure 11.

The interconnection of the pipeline stage registers includes pipeline multiplexers 116, 118 and 120, one of the multiplexers being situated between each pair of adjacent stage registers. Each multiplexer has an input from decoder 60 and from each preceding pipeline stage register, and provides its output to the succeeding stage registers and directly out of instruction pipeline 62, again as represented schematically by "exit" 88. Each multiplexer is selectively switched to provide only one of its inputs to the next subsequent stage. For example, multiplexer 118 selectively couples either decoder 60, register 92, or register 94, to register 96.

Multiplexer switching is controlled by collision detection and arbitration circuit 122 in cooperation with information in each control word. Arbitration circuit 122 receives inputs from decoder 60 and each of the pipelin stag r gisters. In turn, arbitration circuit 122 generates latch data outputs to the pipeline stag registers. In connection with stag registers 94, 96 and 98, arbitration circuit 122 also provides control signals to the associated multiplexers 116, 118 and 120 that l ad the pipeline stage registers.

Through its connections to the stage r gisters, multiplex rs and d coder 60, arbitration circuit 122 resolves cont ntions for pipeline stage r gisters according to the flow chart in Figure 5.

If instruction pipeline 62 is empty, the control word is immediately provided to the selected pipeline stage register. If not, a determination is made as to the dependency of the control word upon any preceding control word in the pipeline. In other words, there is an inquiry as to whether the incoming control word will be dependent upon the results from any currently executing control words. The inquiry is accomplished by checking each of stage registers 92, 94, 96 and 98 as to whether the register will be altered by the control word currently residing in the register. For example, suppose a control word entering the pipeline at address stage 80 (register 92) is destined to use a particular data register from among general purpose registers 54 to calculate an address. Further, suppose that a preceding instruction at execution stage 84 (register 96) involves storing data to that same particular data register. In this event, the entering control word is not loaded into register 92 until the operation in register 96 is complete.

Upon such completion, or if no dependency is found, the selected register is scanned to determine that it will be available on the next machine cycle, in the sense that no other control word already in the pipeline is scheduled to enter the selected stage register during the next cycle.

Given the indication that the stage is available, it becomes necessary to determine whether the control word, upon entering the destination stage, would bypass or "leap frog" ahead of any preceding control words already executing in the pipeline. If not, the control word is immediately loaded into the appropriate stage register.

However, if the control word would bypass any preceding control words when entering the appropriate stage, a further determination is necessary; namely, whether the control word, upon bypassing any preceding control word, would alter conditions used or to be used by any such preceding control word.

As an example, suppose that a control word is to enter the pipeline at stage 86 (register 98) to store data, e.g. from register 72. Further, suppose that a preceding control word, i.e. one previously entering the pipeline, currently resides in operand stage 82 and will reside in execution stage 84 to perform an operation on data that will alter the contents of register 72. In this event, the later entering control word is not loaded into result storing stage 86 until the preceding control word has completed its operation in x cution stage 84. Once the preceding control word has completed its operation, the later entering control word is loaded into the result storing stag .

Further logic resolves contentions to avoid collisions, .g. among control words competing for access to pipeline stag register 98 fr m th decoder and from the preceding pipeline stag registers. Mor particularly, a control word from stag register 96 is favored with respect to the other inputs to multiplexer 120, followed by the output of stage register 94, the output of stage register 92 and the output of decoder 60.

A control word 124 is shown in Figure 6, and includes binary bits and fields of bits corresponding to the bit positions and segments in the pipeline stage registers. A map field 126 of the control word includes four bit positions labeled A, O, X and S, respectively. A condition code bit 128 is provided between map field 126 and a plurality of control fields. For convenience, a single control field is shown in connection with each of the pipeline stage registers, including an address generation control field 130, an operand loading control field 132, an execution control field 134 and a result storing control field 136. Program counter, immediate address and immediate data fields are shown at 135, 137 and 139, respectively. Finally, the control word includes a stage valid bit 138.

Map field 126 is related to the control fields, in that it indicates which types of control fields are present in the control word. For example, a binary "1" at bit position "A" in the map field indicates that control word 124 includes at least one address generation control field. Accordingly, control word 124 is destined to govern an address calculating operation on data as it progresses through the instruction pipeline. As another example, a binary "O" in the "X" bit position of map field 126 indicates that control word 124 has no execution control fields. Control word 124 is destined to proceed through the instruction pipeline without governing any execution functions on data. Thus, control word 124 does not enter execution stage 84.

The map field of each control word routes the control word, by predetermining the pipeline stages that the control word will occupy as it progresses through pipeline 62. A value of "1111" routes the control word sequentially through all four pipeline stages, while a value of "1001" routes the control word only through address generation stage 80 and result storing stage 86.

While certain functions performed on data require all four types of operations, numerous functions do not. In connection with these latter functions, the selective routing of the control word only through stages that "use" the control word substantially enhances instruction pipeline efficiency. Instructions can be channeled through the instruction pipeline with fewer machine cycles, and the machine cycles saved become available to accept further control words. The s lectiv routing involves a hardware response to the map field bits, and thus adds virtually nothing to measurabl processing time.

Figures 7-10 illustrat portions f control word

124 in greater detail, and facilitate an und rstanding of how th se portions relate to th hardwar illustrated in Figure 2. Addr ss g n rating control field 130 includes two multiplexer subfields 140 and 142 respectively concerning address multiplexers 64 and 74. More particularly, the content of subfield 140 determines the source of data from among the potential sources shown in Figure 2, and provides for a further alternative of forcing binary zeros into multiplexer 64. Five bit positions in subfield 140 provide up to thirty-two alternative sources of data provided to this multiplexer. For example, "00000" forces all zeros into the multiplexer, while "11000" selects one of the registers in general register set 54, whereby the address value from the selected register is provided to multiplexer 64. In a similar manner, the value in subfield 142 determines the source of data provided to multiplexer 74.

The subfields "W" and "D" respectively determine the length (number of bits) of the data word provided to multiplexer 74, and whether or not the contents of immediate address field 137 of the control word are to be added to the multiplexer inputs.

Finally, the value contained in a memory destination subfield 144 controls multiplexer 67. Typically the output of effective address unit 66 is a calculated memory address, to designate the destination of data to memory from stage 86. This output also determines the memory address from which data is provided to line 36 via interface 34. Further, however, the output of effective address calculating unit 66 can be provided to other destinations, e.g. to load program counter 48, to one of general purpose registers 54 through multiplexer 52, or to the multiplexer 67 for storing data identified by the address to memory via memory interface 34.

The operand loading field (Figure 8) includes a subfield 146 that determines the source of data provided to multiplexer 38, a subfield 148 that determines the source of data for multiplexer 40, and a subfield 150 for determining the destination of data from main memory 20 to path 36 via I/O memory interface 34. As for destinations, the destination of data from multiplexer 38 is register 44, and data from multiplexer 40 always is provided to register 46. Based on the value in subfield 150, data from memory can be provided (alternatively) to status register 56, condition code register 58, or to one of the general purpose registers 54 via multiplexer 52.

Execution unit 42 receives input operands from registers 44 and 46. Accordingly, no source determining subfield is necessary for the execution control field. A function subfi ld 152 determin s the one of several available functions which execution unit 42 is to perform on the operands provided by registers 44 and 46, and includes five binary bits for selecting a function, .g. adding th operands, subtracting one from the other, performing logic functions such as "AND", "OR", or "Exclusiv OR", and shifting register contents by predetermined amounts, as well as outputting all z ros. A data type subfield 154 and length subfield 156 respectively determine whether data is logical, arithmetic or binary coded decimal, and whether the operation is on a single bit, a single word of two bytes or a double word of four bytes. The binary value in a destination field 158 determines where the execution unit output is sent. Among possible destinations are one of the general purpose registers, the status register, the condition code register and register 72, the holding latch to main storage memory 20.

Storage control field 136 includes a source subfield 160 that determines the source of stored data, e.g. either register 68, register 72 or a forced storage of all zeros. A length subfield 162 determines the length of stored data (single byte verses two-byte or four-byte word).

Each of the control fields is valid in, and only in, its respective one of the pipeline stages. Address generation control field 130, for example, governs the calculation of an effective address while the control word resides in address generation register 92. Control field 130, however, serves no useful purpose in any of the subsequent pipeline stage registers, although the control word may contain other types of control fields and temporarily reside in subsequent stage registers.

Accordingly, each succeeding stage register is configured to provide no bit positions for stage-specific information used in preceding stages of the pipeline. As seen in Figure 11, the stage-specific information includes not only the control fields, but also the map field, where each bit position is deleted after its related pipeline stage. Thus, selected bit positions and register segments are removed from succeeding stage registers, to delete from each control word certain data no longer useful in governing data operations as the control word progresses through the pipeline.

Figure 12 illustrates an alternative configuration instruction pipeline 166 in which assembly level instructions address a control store array 168. The control store array provides the pipeline input in lieu of decoder 60. The pipeline includes multiple stages, three of which are shown at 170, 172 and 174 which represents the "Nth" stage. Multiplexers 176 and 178 interconnect the stages, with a pipeline "exit" 180 indicated in broken lines. As compared to using decoder 60, use of an array such as 168 reduces processing speeds, but affords greater flexibility in terms of compatibility with a wider range of computer architectures.

Stage 170 includes a map segment 182 with bit positions 1-N, control fields 1-N and a stage valid bit position V. Subsequent stages exemplify the information deletion feature of the invention, in particular final stage 174 having a single bit map segment N and a

single control field lab led "FIELD N" along with the stage valid bit position. Each rightward pointing arrow represents an input to its associat d multipl x r from array 168. The leftward pointing arrows represent inputs to the multiplexer from preceding stages. Every combination of one or more pipeline stages is available in connection with routing each control word through the pipeline.

Thus, in accordance with the present invention a multiple stage instruction pipeline is configured for efficient use of available machine cycles, in reducing the cycles required for certain microinstructions to reside in the pipeline. Microinstructions are selectively routed through the pipeline, each microinstruction residing only in selected pipeline stages, based upon the data operation control fields in the microinstruction. A mapping field in each microinstruction facilitates identifying the data operations to be executed as each microinstruction progresses through the instruction pipeline. Finally, arbitration logic is provided to avoid collisions among micro-instructions contending for access to one of the pipeline stages, and further to insure that a control word can skip ahead of another control word formerly preceding it, but only under conditions of nondependency and where the reversal of order can not cause error.

Claims

1. An apparatus for governing the execution of operations upon data in a digital processing device, including:
   a pipeline including a plurality of interconnected stages, each stage being dedicated to one of a plurality of data operations executed on data in a digital processing device;
   a pipeline input for receiving a plurality of digital control words into the pipeline for governing the execution of the data operations as the digital control words progress through the pipeline;
   each of the digital control words including control information corresponding to at least one selected operation of the data operations and mapping information indicating each selected data operation to which the control information corresponds;
   each of the digital control words further governing the performance of the at least one selected data operation when the control word resides in a particular one of the pipeline stages dedicated to the selected data operation;
   means coupling the pipeline input and a pipeline exit to each one of the pipeline stages to enable each of the control words to occupy only selected ones of the pipeline stages as the control word progresses through the pipeline; and
   a control word routing means for s lectively and individually routing each said control word through the pipeline via only selected pipelin stages, the sel cted pip lin stages for ach said control word depending upon th mapping information of the control word.

2. The apparatus of Claim 1 wherein:
   the selected pipeline stages for each data word are the stages dedicated to the data operations identified in the mapping information of the digital control word.

3. The apparatus of Claim 2 wherein:
   all of the pipeline stages are connected to one another in a predetermined sequence, and each of the digital control words progresses through the pipeline according to said predetermined sequence while bypassing pipeline stages other than said selected pipeline stages.

4. The apparatus of Claim 3 further including:
   a plurality of multiplexers coupled between successive ones of the pipeline stages, each of the multiplexers having inputs coupled to all preceding pipeline stages in the predetermined sequence of pipeline stages.

5. The apparatus of Claim 4 wherein:
   each of the multiplexers selectively inhibits control fields valid in a preceding pipeline stage from occupying a next subsequent pipeline stage, thereby to selectively delete control fields of each control word as the control word progresses through the pipeline.

6. The apparatus of Claim 1 wherein:
   the control information of each control word resides in at least one of a plurality of multibit fields, each of the multibit fields being valid in only one of the pipeline stages.

7. The apparatus of Claim 1 wherein:
   the mapping information of each digital control word resides in a mapping field of the control word including one bit position corresponding to each of the pipeline stages.

8. The apparatus of Claim 1 further including:
   an arbitration means for resolving conflicts among control words attempting to enter the same one of the pipeline stages.

9. The apparatus of Claim 1 wherein:
   ach of the pipeline stages is a r gister.

10. The apparatus of Claim 1 wher in:
   ach control word further includ s a stage valid field for indicating with respect to each of th pi-

peline stages the presence of a valid control word in the pip line stage.

11. The apparatus of Claim 1 wherein: the digital processing device includes a condition code register, and each of the control words further includes a condition code bit position for indicating whether the control word will alter the condition code register as the control word progresses through the pipeline.

12. In a data processing device including a pipeline having a plurality of pipeline stages; each pipeline stage dedicated to one of a plurality of data operations executed on data in the processing device, wherein the data operations are governed by a plurality of digital control words as the control words progress through the pipeline, a process for executing the data operations, including the steps of:
making a determination, for each of the digital control words, of selected ones of the data operations for which the control word includes control information; and
responsive to the determination, selectively routing the digital control word through the pipeline in an individual sequence of the pipeline stages, whereby the digital control word, as it progresses through the pipeline, occupies only the pipeline stages dedicated to the selected data operations.

13. The process of Claim 12 wherein: the determination step includes providing a map field in each of the digital control words, each map field including mapping information that identifies each of the data operations for which the control word includes control information.

14. The process of Claim 12 including the further step of:
resolving conflicts among digital control words concurrently attempting to enter any one of the pipeline stages.

15. The process of Claim 14 wherein: the step of resolving conflicts includes providing a valid bit identification in each digital control word that is set whenever a valid control word resides in one of the pipeline stages, and preventing any other digital control word from entering a given pipeline stage when a valid bit is set in the given pipeline stage.

16. The process of Claim 13 including the further step of:
providing a condition code bit in each control word indicating, if set, that the control word will change a condition code register of the data processing device as the control word progresses through the pipeline.

17. The process of Claim 12 including the further step of:
making a determination, for each digital control word, as to whether the digital control word depends upon a result of a preceding control word currently executing in the pipeline and, if so, delaying the routing of the digital control word until the preceding control word has completed said execution.

18. The process of Claim 12 including the further steps of:
determining, for each digital control word, whether said selectively routing of the digital control word will cause the digital control word to bypass a currently executing control word already in the pipeline;
if so, determining whether the digital control word, if it bypasses the currently executing control word, would alter a condition of the currently executing control word; and
if so, delaying said routing of the digital control word until the currently executing control word completes said execution, thereby either to delay or prevent said bypass.

Fig.-2

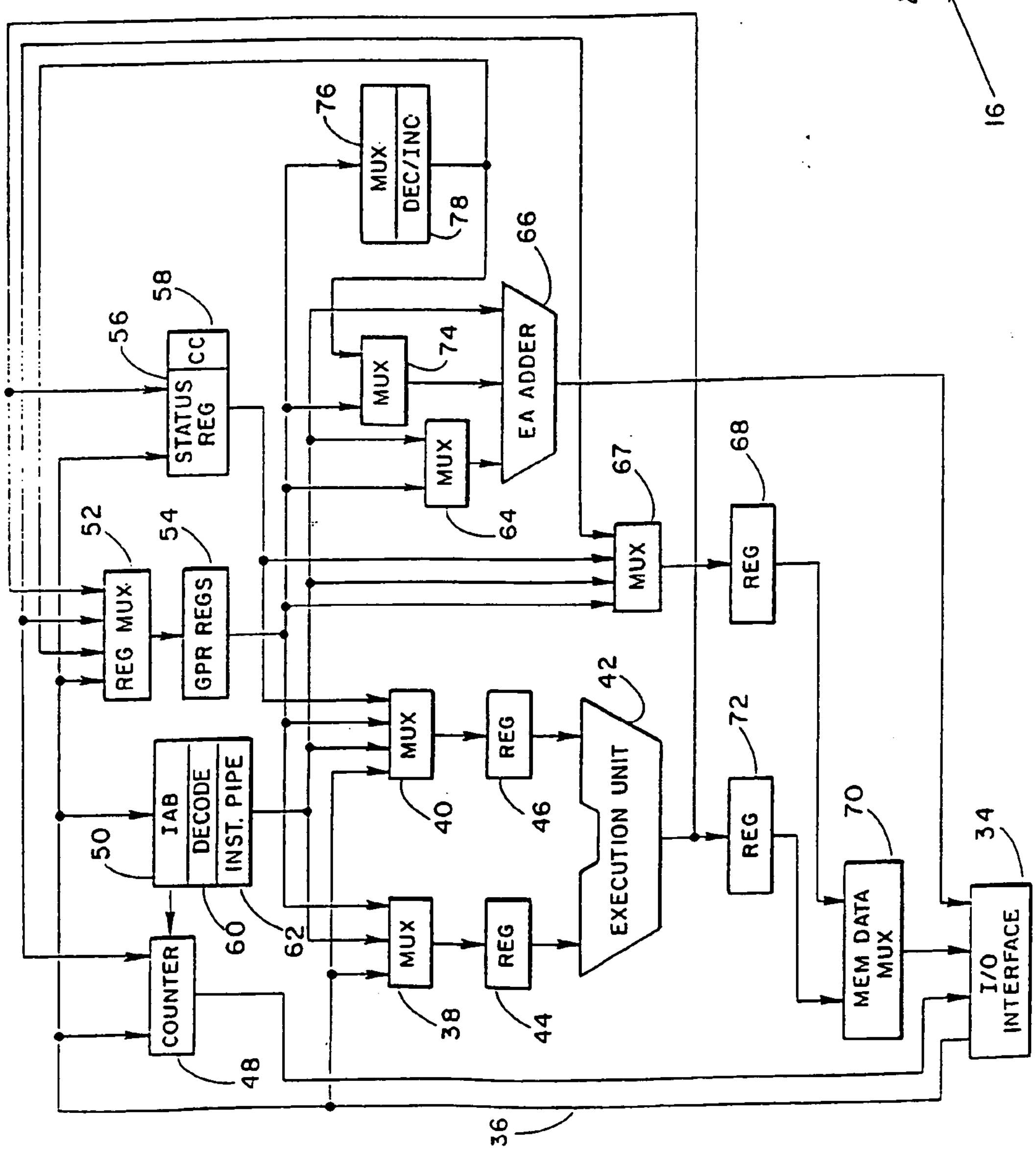

REG MUX
GPR REGS
STATUS REG
CC
IAB
DECODE
INST. PIPE
COUNTER
MUX
DEC/INC
EA ADDER
MUX
MUX
MUX
REG
MUX
REG
MUX
REG
EXECUTION UNIT
REG
REG
MEM DATA MUX
I/O INTERFACE
36
38
40
42
44
46
48
50
52
54
56
58
60
62
64
66
67
68
70
72
74
76
78
34

Fig.-1

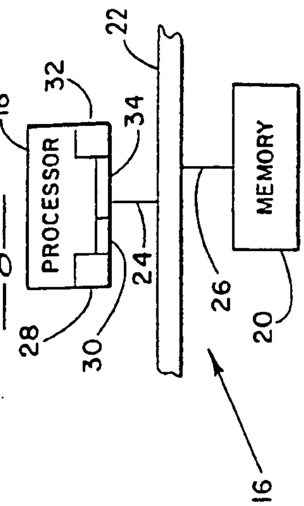

PROCESSOR
MEMORY
16
18
20
22
24
26
28
30
32
34

Fig.-3

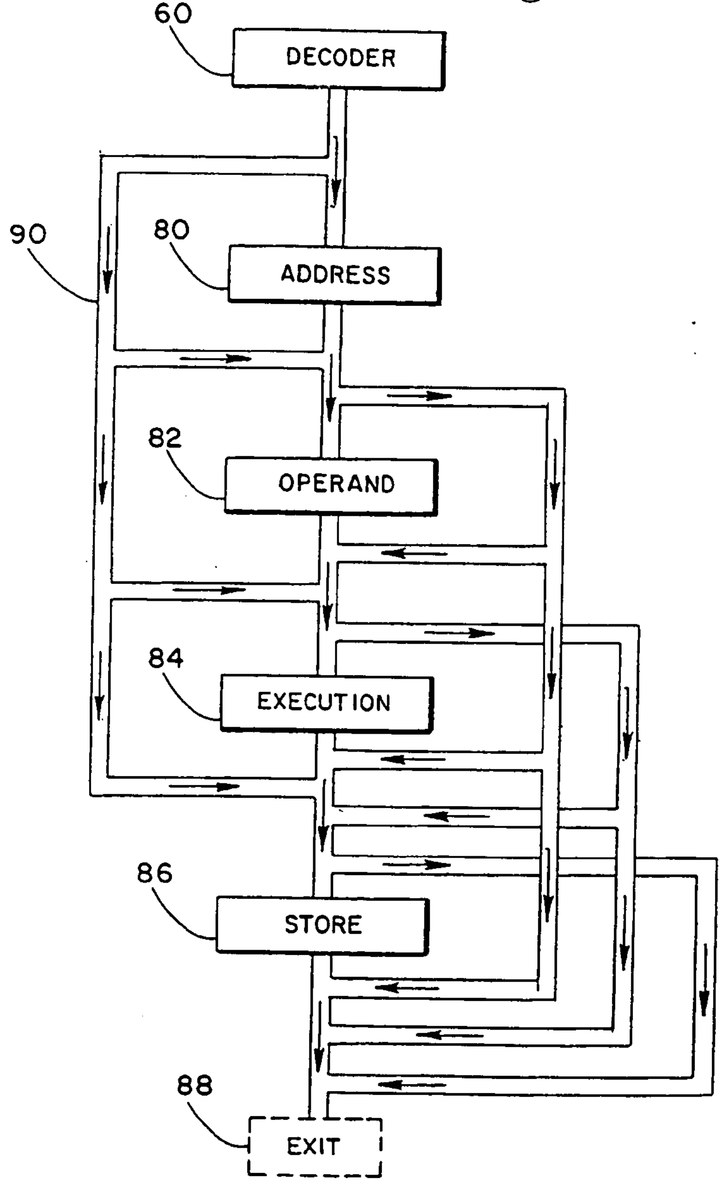

60
DECODER
90
80
ADDRESS
82
OPERAND
84
EXECUTION
86
STORE
88
EXIT

Fig.-4

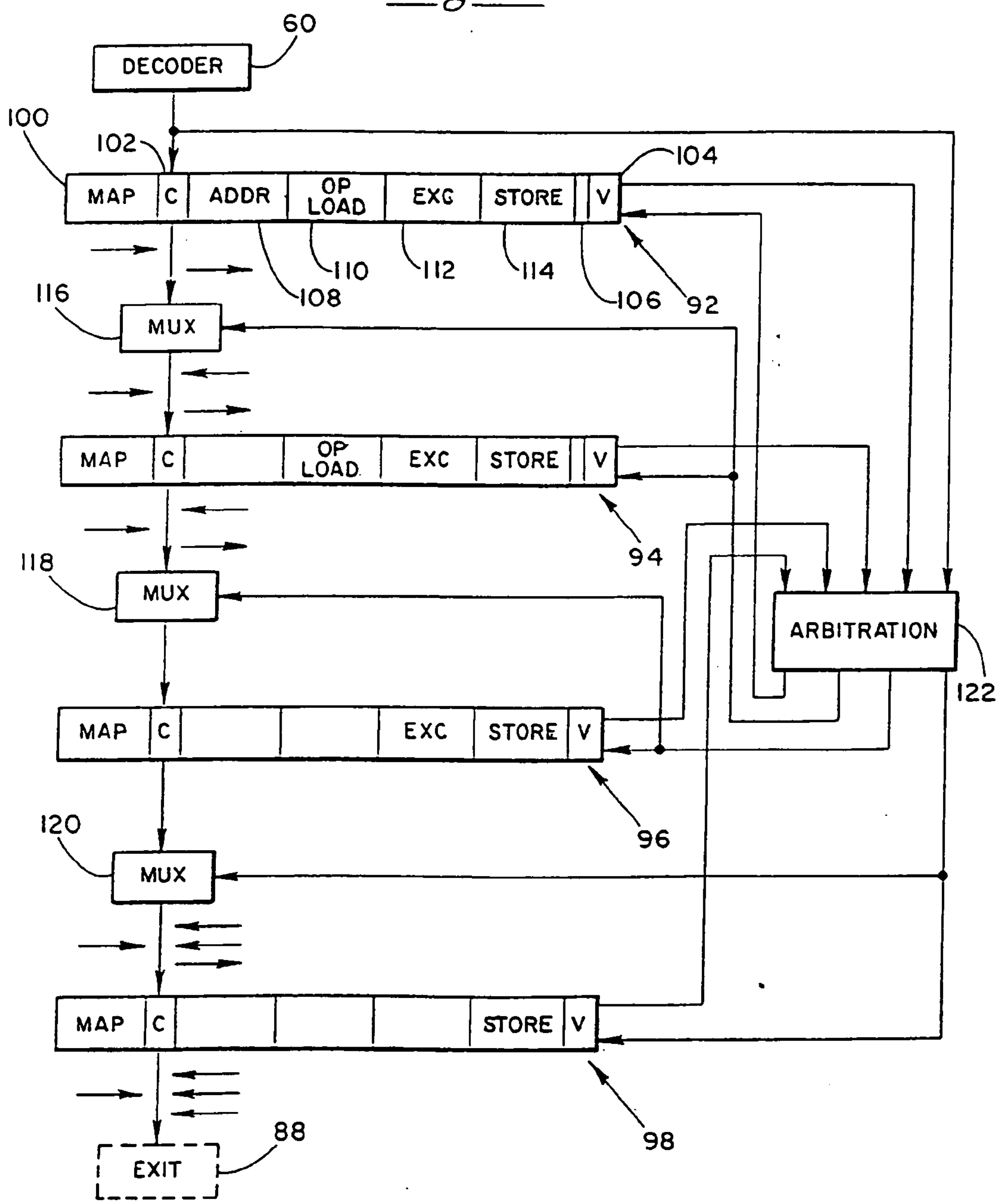

60
DECODER
100
102
104
MAP
C
ADDR
OP LOAD
EXC
STORE
V
108
110
112
114
106
92
116
MUX
MAP
C
OP LOAD
EXC
STORE
V
94
118
MUX
ARBITRATION
122
MAP
C
EXC
STORE
V
96
120
MUX
MAP
C
STORE
V
98
88
EXIT

Fig.-5

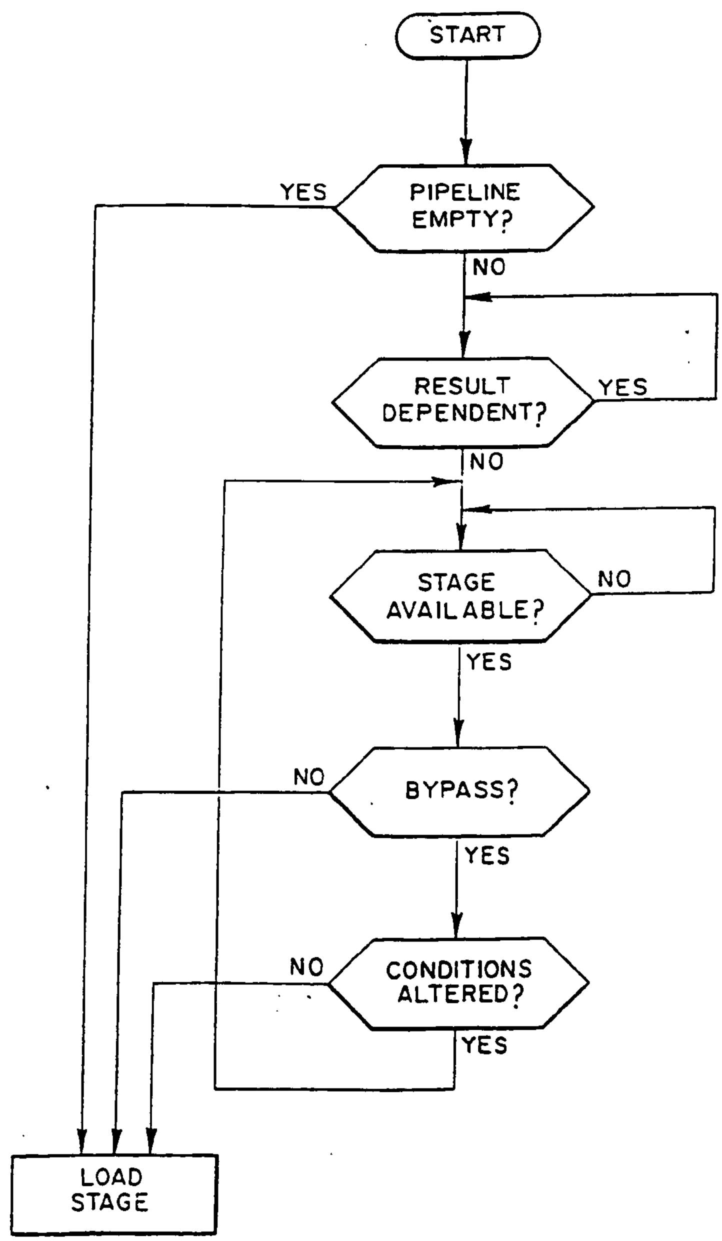

START
PIPELINE EMPTY?
YES
NO
RESULT DEPENDENT?
YES
NO
STAGE AVAILABLE?
NO
YES
BYPASS?
NO
YES
CONDITIONS ALTERED?
NO
YES
LOAD STAGE

Fig.-6

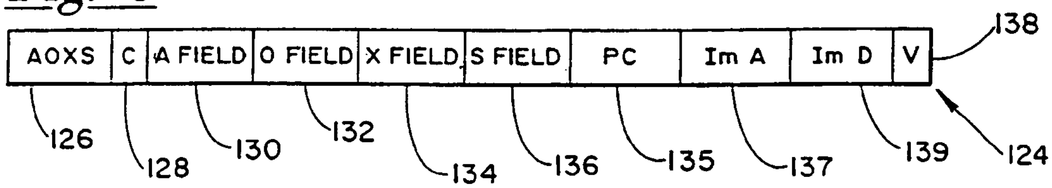


Fig.-7

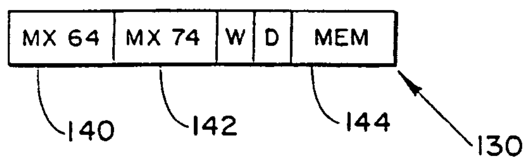


Fig.-8

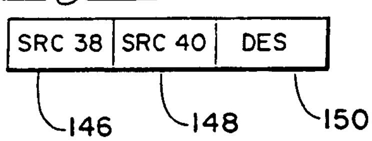


Fig.-9

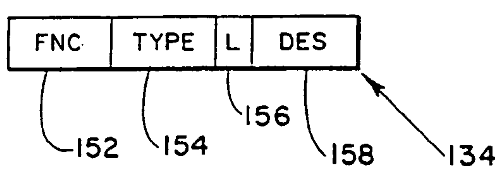


Fig.-10

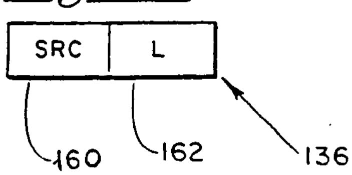


Fig.-11

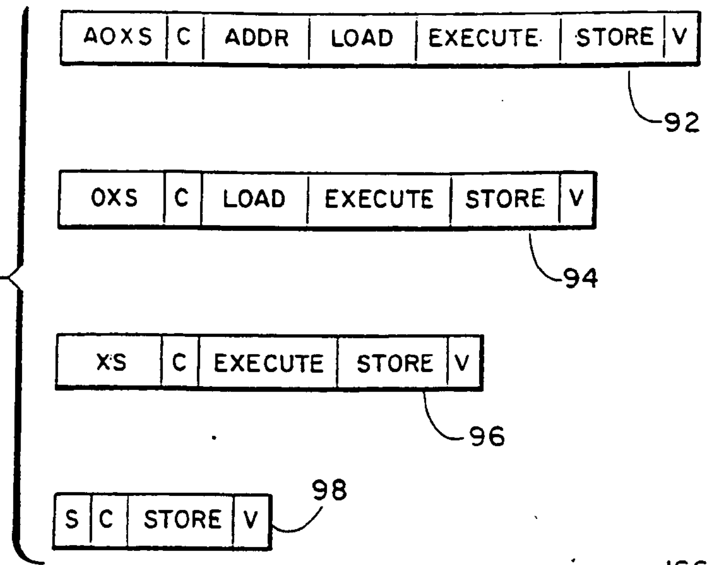
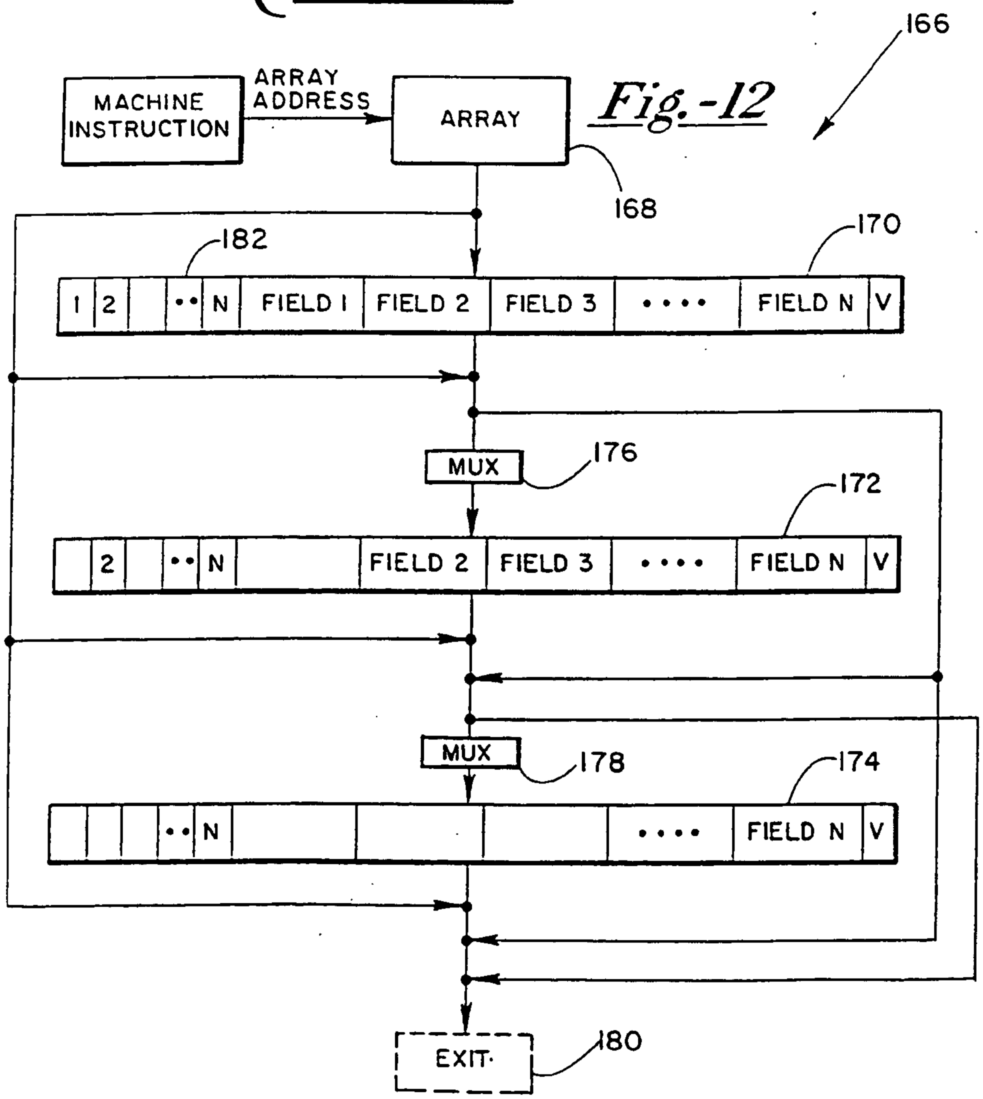
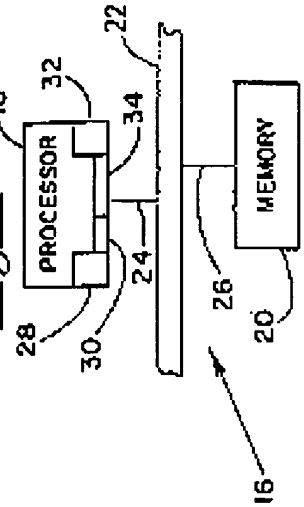

AOXS
C
ADDR
LOAD
EXECUTE
STORE
V
92
OXS
C
LOAD
EXECUTE
STORE
V
94
XS
C
EXECUTE
STORE
V
96
S
C
STORE
V
98

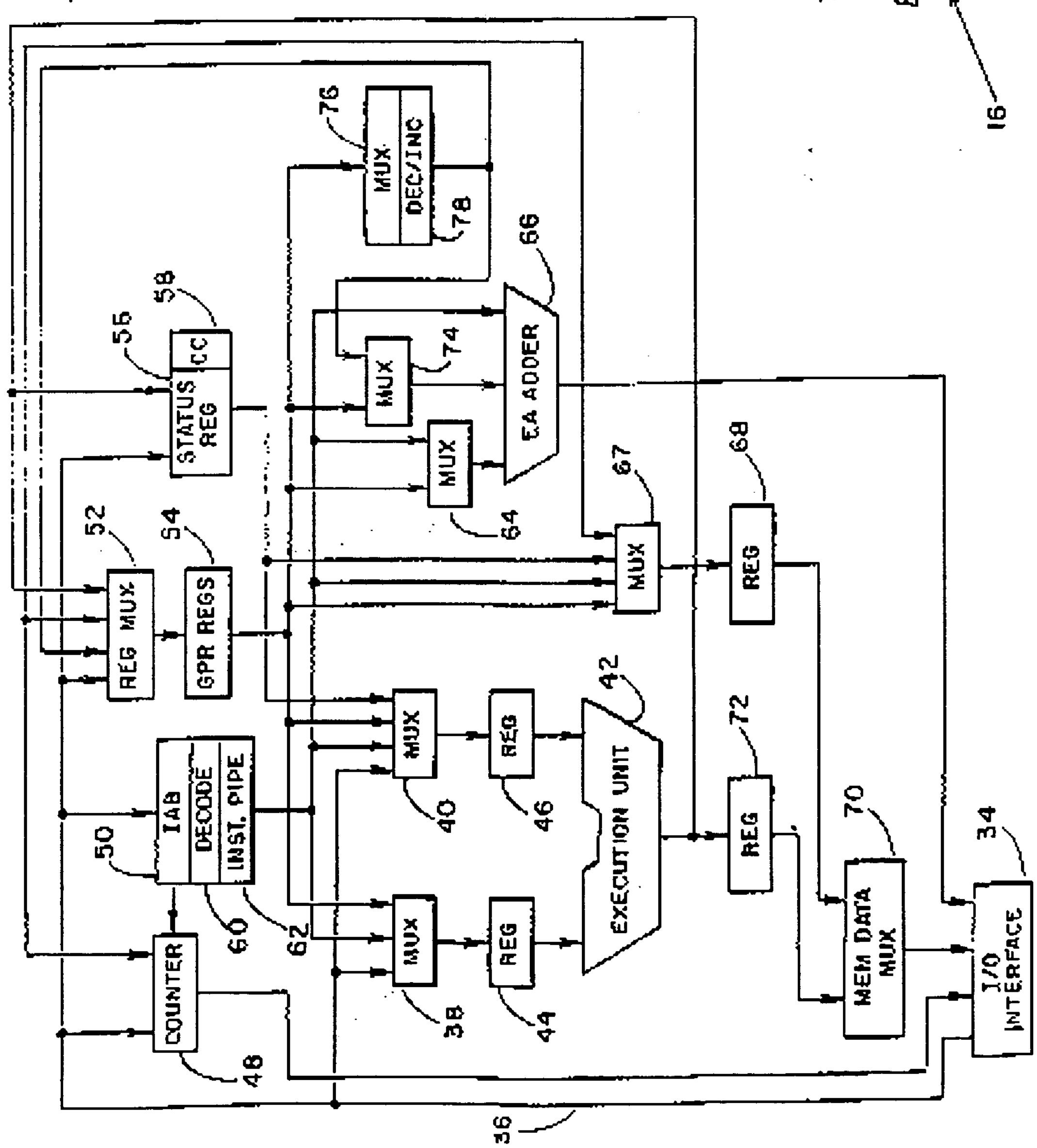
Fig.-12


166
MACHINE INSTRUCTION
ARRAY ADDRESS
ARRAY
168
182
170
1
2
••
N
FIELD 1
FIELD 2
FIELD 3
• • • •
FIELD N
V
MUX
176
172
2
••
N
FIELD 2
FIELD 3
• • • •
FIELD N
V
MUX
178
174
••
N
• • • •
FIELD N
V
EXIT
180



PROCESSOR
MEMORY
18
32
34
22
28
30
24
26
20
16

Fig.-1



MUX
DEC/INC
76
78
66
EA ADDER
74
MUX
55
58
STATUS REG
CC
MUX
64
52
REG MUX
54
GPR REGS
67
MUX
68
REG
IAB
DECODE
INST. PIPE
50
60
62
MUX
40
REG
46
42
EXECUTION UNIT
72
REG
70
MEM DATA MUX
34
I/O INTERFACE
COUNTER
48
MUX
38
REG
44
36

Fig.-2

Fig.-3

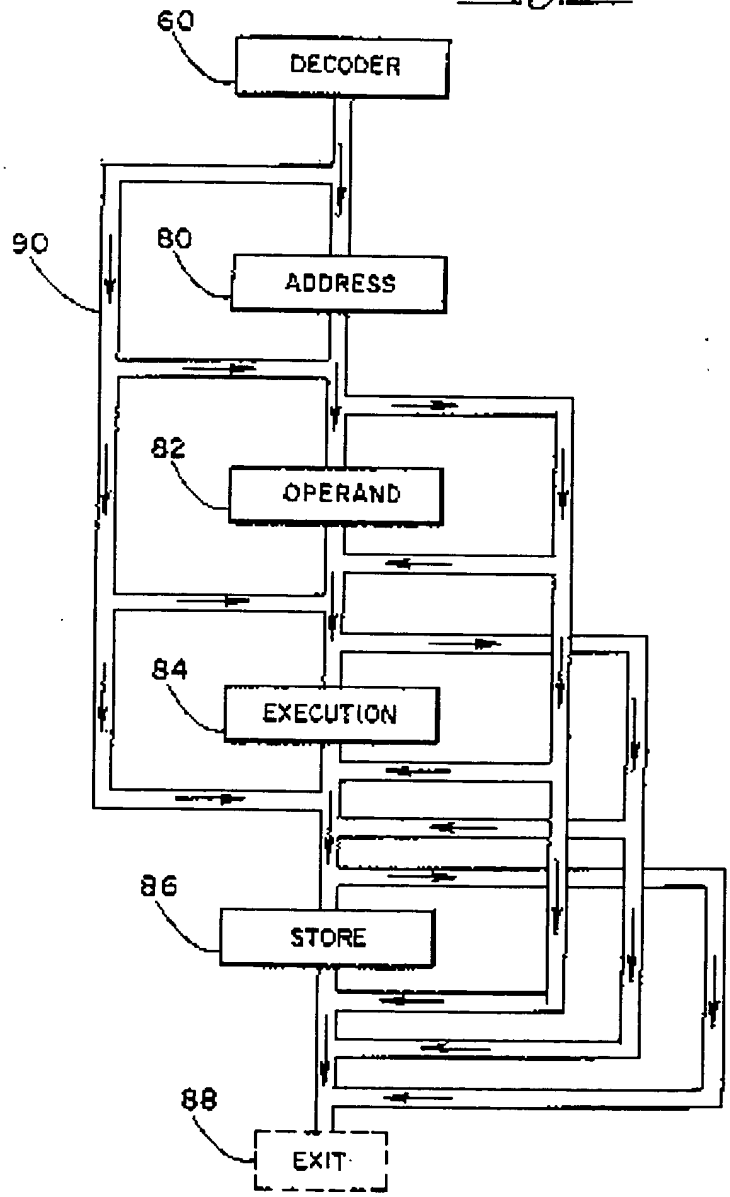

60
DECODER
90
80
ADDRESS
82
OPERAND
84
EXECUTION
86
STORE
88
EXIT

Fig.-4

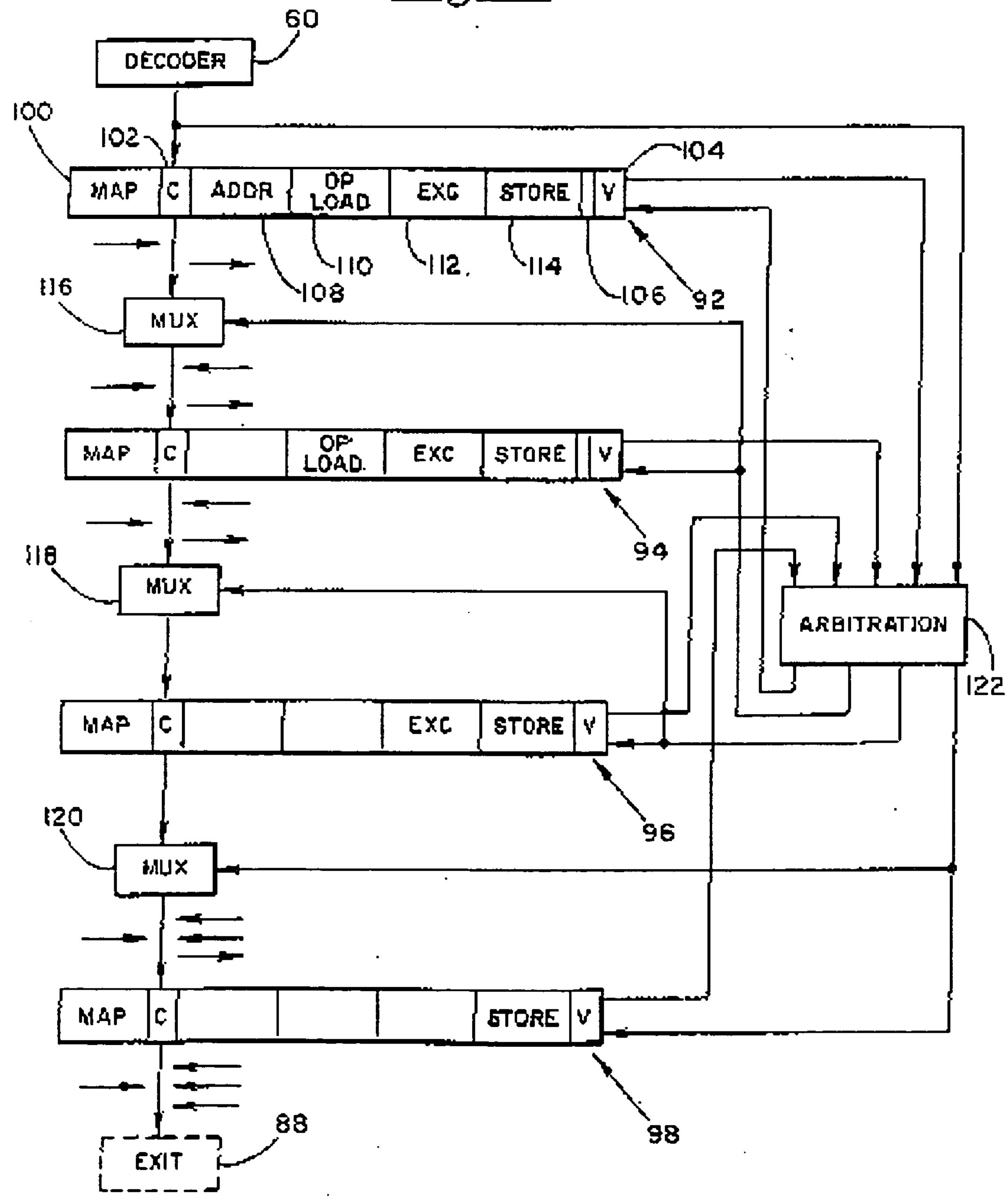

60
DECODER
100
102
104
MAP
C
ADDR
OP LOAD
EXC
STORE
V
108
110
112
114
106
92
116
MUX
MAP
C
OP LOAD
EXC
STORE
V
94
118
MUX
ARBITRATION
122
MAP
C
EXC
STORE
V
96
120
MUX
MAP
C
STORE
V
98
88
EXIT

Fig.-5

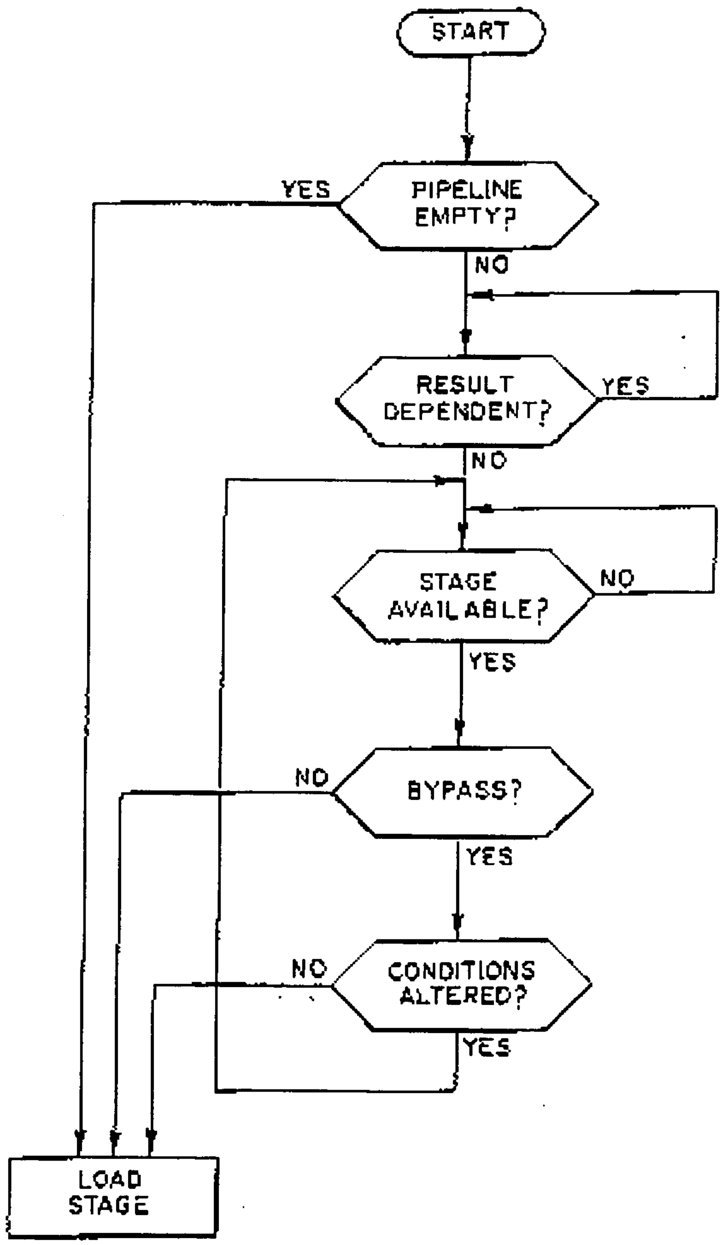

START
PIPELINE EMPTY?
YES
NO
RESULT DEPENDENT?
YES
NO
STAGE AVAILABLE?
NO
YES
BYPASS?
NO
YES
CONDITIONS ALTERED?
NO
YES
LOAD STAGE

Fig.-6

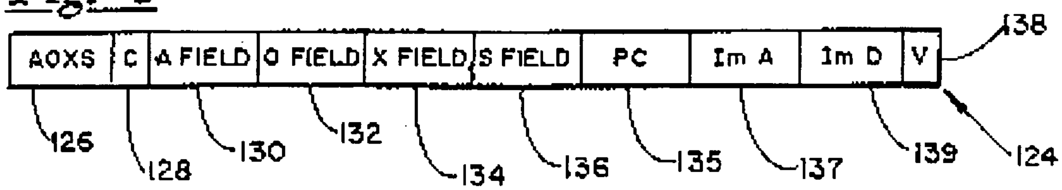


Fig.-7

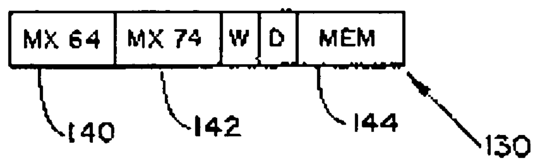


Fig.-8

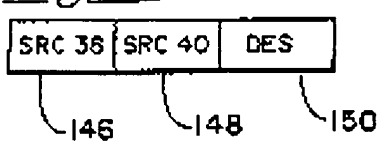


Fig.-9

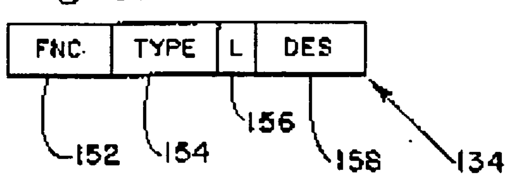
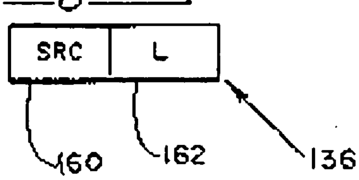


Fig.-10



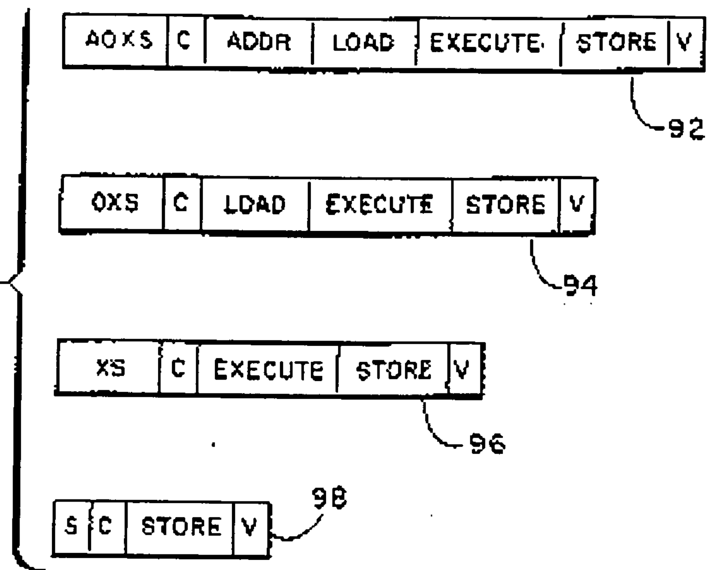

AOXS
C
ADDR
LOAD
EXECUTE
STORE
V
92
OXS
C
LOAD
EXECUTE
STORE
V
94
XS
C
EXECUTE
STORE
V
96
S
C
STORE
V
98

Fig.-11

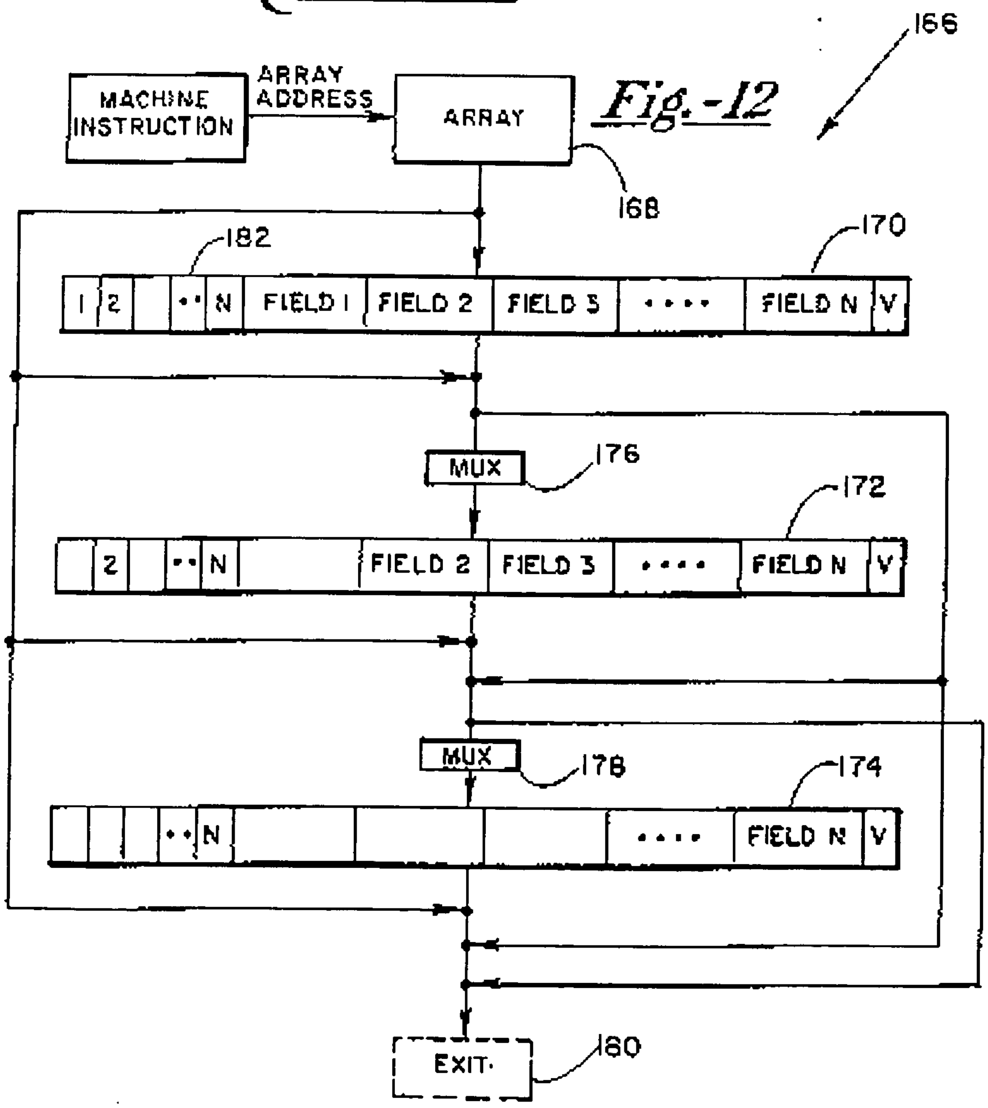

166
MACHINE INSTRUCTION
ARRAY ADDRESS
ARRAY
168
182
170
1
2
N
FIELD 1
FIELD 2
FIELD 3
FIELD N
V
MUX
176
172
2
N
FIELD 2
FIELD 3
FIELD N
V
MUX
178
174
N
FIELD N
V
EXIT
180

Fig.-12